



Ksh Basics & Syntax

Basic Syntax

<code>#!/bin/ksh</code>	Shebang line, specifies the interpreter for the script.
Variable Assignment	<code>variable=value</code> (No spaces around <code>=</code>) <code>readonly variable=value</code> (Makes the variable read-only)
Variable Usage	<code>\$variable</code> or <code> \${variable}</code> (for clarity, especially with concatenation)
Command Substitution	<code>\$(command)</code> or <code>command</code>
Comments	<code># This is a comment</code>
Exit Status	<code>\$?</code> - Access the exit status of the last executed command (0 is success).
String Concatenation	<code>string="\$var1\$var2"</code>

Input/Output

Reading Input	<code>read variable</code> (Reads a line from standard input)
Printing Output	<code>print "message"</code> or <code>echo "message"</code> (Prints to standard output)
Redirecting Output	<code>command > file</code> (Redirect standard output to file, overwriting) <code>command >> file</code> (Append standard output to file) <code>command 2> file</code> (Redirect standard error to file) <code>command >&gt; file</code> (Redirect both standard output and error to file)
Here Documents	<code>cat << EOF text... EOF</code> (Passes multi-line text to a command)

Variables

Types of Variables	String variables (default), integer variables (using <code>integer</code>), arrays (using <code>typeset -a</code>)
Exporting variables	<code>export variable</code> (makes the variable available to subprocesses)
Unsetting a variable	<code>unset variable</code>
Special variables	<code>\$0</code> (Script name), <code>\$1, \$2, ...</code> (Arguments), <code>#\$</code> (Number of arguments), <code>\$\$</code> (Process ID)

Control Flow

Conditional Statements (if/then/else)

Loops

```
if [ condition ]; then
    commands
elif [ condition ]; then
    commands
else
    commands
fi
```

Example:

```
if [ "$var" = "value" ]; then
    print "Var is value"
fi
```

```
For Loop      for variable in list; do
                commands
            done
```

Example: `for i in 1 2 3; do
 print $i; done`

```
While Loop    while [ condition ]; do
                commands
            done
```

Example:

```
i=0
while [ $i -lt 5 ]; do
    print $i
    i=$((i+1))
done
```

```
Until Loop    until [ condition ]; do
                commands
            done
```

```
Select Loop   select variable in list; do
                commands
                break #Important to add
                break
            done
```

Case Statements

```
case $variable in
    pattern1) commands ;;
    pattern2) commands ;;
    *) commands ;;
esac
```

Example:

```
case $option in
    -a) print "Option A" ;;
    -b) print "Option B" ;;
    *) print "Invalid option" ;;
esac
```

Functions and Arrays

Functions

Defining a Function

```
function_name() {  
    commands  
}
```

or

```
function function_name {  
    commands  
}
```

Calling a Function

```
function_name arg1 arg2
```

Returning Values

```
return value
```

 (Returns an exit status, 0-255. Use `print` for other values.)

Local Variables

```
local variable=value
```

 (Declares a variable with local scope)

Arrays

Declaring an Array

```
typeset -a array_name
```

Assigning Values

```
array_name[0]=value1 array_name[1]=value2
```

Accessing Elements

```
echo ${array_name[0]}
```

Array Length

```
echo ${#array_name[*]} or echo  
${#array_name[@]}
```

All Elements

```
echo ${array_name[*]} or echo  
${array_name[@]}
```

String Manipulation and Built-in Commands

String Manipulation

Substring Extraction

```
echo ${variable:offset:length}
```

String Length

```
echo ${#variable}
```

Pattern Replacement

```
echo ${variable/pattern/replacement}
```

 (Replaces first occurrence)

```
echo ${variable//pattern/replacement}
```

 (Replaces all occurrences)

Case Conversion

```
echo ${variable^^}
```

 (Uppercase)

```
echo ${variable,,}
```

 (Lowercase)

Built-in Commands

```
pwd
```

Print working directory

```
cd directory
```

Change directory

```
ls
```

List files and directories

```
mkdir directory
```

Create directory

```
rm file
```

Remove file

```
cp source destination
```

Copy file

```
mv source destination
```

Move or rename file

```
test expression or [  
expression ]
```

Evaluate expression (used in conditionals)